Theoretical Science Group

理論科学グループ



部報 235 号
— 新歓コンパ号 —

目 次

年間行事予定 **1**
年間行事予定表 . . . . 【編集部】 1

一般記事 **3**
はじめての ASP . . . . 【Kit】 3
東めた ADSL と dyndns.org によるドメインの運用について . . . . 【Tossy-2】 12
DirectDraw の半透明処理第一回 . . . . 【まめ】 24
酒との上手な付き合い方 . . . . 【植原 洋介 (ue@tkc.att.ne.jp)】 34
TSG 用語の基礎知識 2001 年度版 . . . . 【ぶほ編集部】 36

年間行事予定

年間行事予定表

編集部

年間行事予定表

四月	オリエンテーション
五月	新歓コンパ
六月	このころに秋葉原分科会が開催されます。
七月上旬	分科会終了
八月	夏合宿
九月	駒場祭で作品を発表する人は、少なくともこのころから開発を始めましょう
十月	駒場祭総決起コンパ
十一月	駒場祭
	駒場祭打ち上げコンパ
	役員交代
十二月	クリスマスコンパ。新役員の最初の仕事です。頑張りましょう。
一月	謎のイベント。現時点では詳細をお伝えできません。

各行事紹介

新歓コンパ

文字通り、新入生を歓迎するコンパです。このコンパに参加したからといって TSG に入部しなくてはいけないわけではありません。コンパだけに参加して入部しなくてもまったく問題ありません。

新入生の参加費は安価なので、ふるってご参加ください。コンパでは TSGer の自己紹介が行われます。個性に富んだ TSGer の一面を見ることができるでしょう。

秋葉原分科会

皆で日本最大の電気街、秋葉原へ出かけます。詳しい内容は分科会紹介の秋葉原分科会の所を読んでください。

夏合宿

八月の初めに行われます。今年は 3 泊 4 日@箱根を予定しています。箱根には大変多くの観光スポットがあるので 4 日のうち 1 日はそれらへ出かけることになるでしょう。宿では各々が思い思いの事をしますが、多くの人は持ち込まれた多数のゲームや漫画に興じます。

TSGer の顔を覚え、親しくなるよい機会です。新入生はぜひ参加してください。

総決起コンパ

TSG は駒場祭[1]に企画を出典します。それに向けて総決起集会をするわけです。各々が駒場祭で実現することを発表します。大法螺を吹くことが推奨されるため、ほら吹きコンパとも呼ばれます。

駒場祭

TSG は駒場祭に 4 つの企画 (ゲームなど) を出展します[2]。そのうちの 1 つは新入生全員で作ります。それぞれの得意分野で実力を思う存分発揮してください。

7 月に出展するものを決め、駒場祭までに完成させます。また、夜間居残りといって、駒場祭の深夜にマシンの盗難防止目的で何人かが学校に残ります。[3]その時に様々な楽しい企画が催されるそうです[4]。これを楽しみにしている人もいるようです。

クリスマスコンパ

クリスマスということで、洒落た雰囲気の料理店で行われます。コンパの初めにシャンパン、最後にケーキが出ます。

追出しコンパ

めでたく卒業する先輩を盛大に追出します。

このほかにも、非公式行事が多数催されます。これらの行事を通じて、先輩方と交流したり、コンピュータ以外の事に興じたりできるのも、TSG の魅力といえるでしょう。

[1]駒場で 3 日間開催される学園祭。
[2]駒場祭委員会に届け出るのが 4 つというだけで、実際の出展数はそれ以上。
[3]まだ企画が出来上がっていないために開発目的で残る人も。
[4]私は去年は夜間居残りに参加していないので詳しい内容は知りません。

一般記事

はじめての ASP

Kit

はじめに

新入生のみなさん、はじめまして。前部長で現本郷会長の Kit です。今流行りの ASP の使い方をそっと教えちゃいます。

ASP とは

「暴れん坊将軍プレミアム」の略。これを所持するものには誰もいない砂浜を白馬に乗ってさながら徳川幕府八代将軍吉宗の如く駆け抜ける権利が与えられると言われている[1]が、手にしたものは数人しかいない。類似品にスタンダード、プロフェッショナルがある。
（民明書房 世界のおかしな略語より）

嘘です。

ASP とは（訂正版）

ASP とは Active Server Pages の略です。どういうものかというと、CGI という言葉を知っている人は似たものだと思ってもらえればいいです。CGI を知らない方は、インターネットの掲示板を思い浮かべてください。そのようなことができるのです。わかりましたか？ こんな説明じゃわかるわけないですね（ぉ まあ、難しく考えずに読んでみてください。

ASP は VBScript（または JScript）で書かれた[2]プログラムです。基本的には HTML ファイ

[1]いや、吉宗が実際にあんなことしてたかどうかは知りませんが。
[2]実はそれだけに限らないのですが、ここでは割愛します。

ルの中のところどころにプログラムが埋め込まれた形になっていて、理解しやすい[3]ため、最近いろんなところで使われています。インターネット通販のサイトか何かで URL の最後が .asp になっているのを見たことはありませんか？ 情報通信辞典 e-Words (http://www.e-words.ne.jp/) でも使われてますね。何、e-Words を知らないって？ それはいけない、すぐに行ってみましょう。とてもいいサイトですよ:-)

話がずれてしまいました。ここではその ASP の書き方をできるだけ簡単に教えていきたいと思います。基本は HTML なので、HTML の基礎程度はわかっておいたほうがいいです。あと、プログラムに関する簡単な知識（関数とか演算子とか）がちょっと必要になるかもです。ASP でできることというと結構多岐にわたるため、何回かに分けての連載になるかもしれませんし、1 回で企画倒れになるかもしれません（ぉ

動作環境

Windows 95/98(SE)/Me/NT 4.0/2000
ちょっと調べてみたところ、上記の OS では動くようです。そのうち Windows98 と Windows2000 については僕自身で動作確認はしてます。

前準備

自分でやったことのある Windows98/2000 についてのみ説明します。ただ、Windows98 の説明に関しては記憶に頼っているところが大きいので多少間違いがあるかもしれません。ご了承ください。

Web サーバーのインストール

ASP は Web サーバー上で動きます。よって、Web サーバーのインストールが必要です。
Windows98/2000 ともに「コントロールパネル」→「アプリケーションの追加と削除」から「Windows ファイル（2000 の場合は Windows コンポーネントの追加と削除）」を選んで、Windows98 なら Personal Web Server(PWS)、Windows2000 なら Internet Information Server(IIS) を探し出してインストールします。Windows98 の場合は、Windows98 の CD-ROM の中の ¥add-on¥pws¥setup.exe からインストールすることもできます。
注：WindowsMe の場合は CD-ROM 内に PWS がありません。Windows98 の CD-ROM からインストールすれば大丈夫です。

[3] Perl よりはわかりやすいと僕は思います。

細かい設定などは ASP を動かす分には必要ありません。これにてインストール終了。

Web サーバーが動いているかどうかの確認

ブラウザを立ち上げ、アドレス欄に「localhost」と入れて Enter を押してください。「サーバーが見つかりません」と出た場合[4]には Web サーバーが動いていません。Windows98 の場合はパーソナル Web マネージャを起動して「開始」を押します。Windows2000 の場合、「管理ツール」→「サービス」を選択し、「World Wide Web Publishing Service」を右クリックして「開始」を選択します。「サーバーが見つかりません」以外（「工事中です」など）が出た場合には問題ありません。

エディタ

ファイルを編集するためにエディタが必要です。Windows 標準付属のメモ帳でも構いません。

ではじめよう。

やっとこさ本題です。

まずは新規にファイルを作成しましょう。作成する場所は C:¥Inetpub¥wwwroot¥にします。ファイル名は何でもかまいませんが、ここでは `test.asp` としておきましょう。`test.asp` をエディタで開いて、以下の内容を入力してください。

———ここから———

```
<% @Language="VBScript" %>
<% Response.Write "Hello, ASP!" %>
```

———ここまで———

入力したら保存しましょう。

では見てみましょう

ブラウザを開いて、アドレス欄に「localhost/test.asp」と入力して Enter を押してみましょう。「Hello, ASP!」と表示されましたか？ 表示されたら成功です。表示されなかったら、上の 2 行がちゃんと入力されているかどうか確認してください。スペルミスはありませんか？

[4]Internet Explorer の場合。Netscape 他ではどうなるかわかりません。

説明

では説明に移ります。

1 行目

「<%」から「%>」までが ASP のプログラム部分です。前にも述べたとおり、ASP は VBScript か JScript で書かれます。この行ではどちらを使って書いているかを明示しています。JScript で ASP を書きたいときには「VBScript」を「JScript」に書き換えればいいわけです。

2 行目

上と同じく「<%」から「%>」までがプログラム部分。何をしているかというと、ブラウザに文字列を表示させています。最初の `Response.Write`、これはブラウザにデータを送信する関数です[5]。使い方は次の通りです。

Response.Write 表示する文字列

ちゃんと Write と文字列の間には半角スペースを空けましょう。文字列は上の例のように「”」（ダブルクォーテーション）で囲みます。

では次のステップへ

これまた以下のものを入力してください。Response.Write の前のスペースの数は適当でいいですよ。下の例ではスペースは 4 つです。

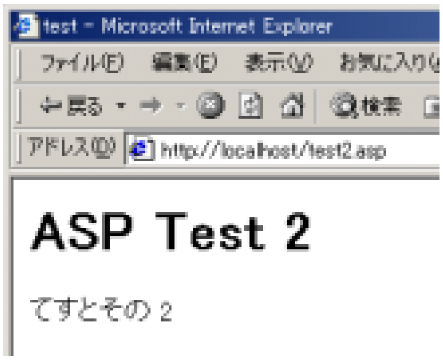
———ここから———

```
<% @Language="VBScript" %>
<%
    Response.Write "<html><head><title>test</title></head>"
    Response.Write "<body>"
    Response.Write "<h1>ASP Test 2</h1>"
    Response.Write "てすとその 2"
    Response.Write "</body></html>"
%>
```

———ここまで———

入力したら保存してブラウザで見てみましょう。

[5]厳密に言えば Response オブジェクトの Write メソッドなのですが。



← のようになってれば OK です。「表示」→「ソース」（IE の場合）でソースを見てみましょう。さっき Response.Write で送信したデータが書かれてますね。このように HTML を送信してやるとブラウザは HTML ファイルとみなして表示してくれます。ソースに改行が入ってないのは、Response.Write で指定した文字列の中に改行文字が含まれていないからです。改行させたい場合は改行文字を出力するよう指定しなくてはいけませんが、その話はまた後で。

他に例をいくつか

例と説明を交互に書いていきます。

```
<% @Language="VBScript" %>
<html>
<head><title>test</title></head>
<body>
<%
    Response.Write "<h1>ASP Test 2</h1>"
    Response.Write "てすとその 2"
%>
</body></html>
```

「<%」と「%>」の間だけが ASP のプログラム部分であり、残りの部分には普通の HTML を書くことができます。動的に生成しなくてもよい部分（タイトルとか）はこの方式で書くのがよいでしょう[6]。むしろ HTML の中に ASP のプログラムが埋め込まれているといった方向で。

```
<% @Language="VBScript" %>
<html>
<head><title>test</title></head>
<body>
<h1><% = "ASP Test 2" %></h1>
てすとその 2
</body></html>
```

「<%」の後にすぐ「=」が来てますね。これは、Response.Write と同じ意味を持ちます。

```
<% = "ASP Test 2" %>
<% Response.Write "ASP Test 2" %>
```

↑の 2 つは同じ結果になる、ということです。

[6] この例は全部静的でいいだろうという突っ込みはなしよ:-)

構文とか

VBScript の構文とかそのあたり。

変数

データの格納場所、変数です。VBScript では変数宣言の必要はありません。どうしても宣言したければ、

Dim 変数名

とします。

次に、変数の型についてです。VBScript では型という概念がありません[7]。同じ変数を整数のために用いたり、文字列のために用いたりできます。というより、Visual Basic のように型指定をするとエラーになってしまいます。

変数の名前に使えるのは半角英数字とアンダーバーです。ただし、大文字と小文字は**区別しません**。注意してください。

演算子

A = B	B を A に代入
A + B	A と B の和
A - B	A と B の差
A * B	A と B の積
A / B	A を B で割った商
A Mod B	A を B で割った余り
-A	A の符号を反転させたもの
A^B	A の B 乗
A ¥ B	A を B で割った商（整数）
A & B	文字列連結
A And B	A と B の論理積
A Or B	A と B の論理和
A Not B	A の論理否定
A Xor B	A と B の排他的論理和
A Eqv B	A と B の論理等価演算

算術及び論理演算子

A = B	A と B が等しい
A <> B	A と B が等しくない
A > B	A が B より大きい
A >= B	A が B 以上である
A < B	A が B より小さい
A <= B	A が B 以下である

比較演算子

演算子はこれで全部ではありません。

[7]これも厳密に言えば違うのですが

条件分岐

条件分岐は以下の用な構文で表されます（[ ] 内は省略可）

```
If 式-1 Then
    処理-1
[ElseIf 式-n Then
    処理-n] ...
[Else
    処理-else]
End If
```

これは次のような意味です。

1. 式-1 を評価して真なら処理-1 を行う。
2. 式-1 が偽だった場合は式-2 を評価し真なら処理-2 を行う。
3. 同様に、式-1 から式-(n-1) が偽だった場合は式-n を評価し真なら処理-n を行う。
4. 式-n まですべて偽だった場合は処理-else を行う。

次のような書き方もできます。ただし 1 行で書ききる場合のみです。

```
If 式-1 Then 処理-1 [Else 処理-2]
```

処理-1 または処理-2 に複数の文を与えたいときは「:」（コロン）で区切ります。

ループ処理

```
For カウンタ変数 = 初期値 To 最終値 [Step 加算値]
    処理
Next
```

これは以下のように動作します。

1. カウンタ変数に初期値を代入する。
2. 処理を行う
3. カウンタ変数に加算値を加える（省略されている場合は 1 を加える）
4. カウンタ変数の値が最終値を超えていない場合は 2 に戻る。
5. 最終値を超えている場合は終了。

ループ処理その 2

```
Do [While 条件]
    処理
Loop
```

または

```
Do
    処理
Loop [While 条件]
```

これは、条件が真である間処理を繰り返します。2 つの違いは、前者がまず条件を評価するのに対して後者はまず処理を行ってそれから条件を評価するという点です。また、While の代わりに Until が使えます。その場合は条件が真になるまで処理を繰り返します。

応用編

応用と言っても、今説明した構文を使ってより「プログラムしてる」気分になろう、というだけです（笑

```
<% @Language="VBScript" %>
<html>
<head><title>ASP 応用</title></head>
<body>
<table border width=450>
<%
    For i = 1 To 9        '9 回のループ（i を 1 から 9 まで変化させる）
        a = i Mod 3       'i を 3 で割った余りを a に代入
        If a = 0 Then     'a が 0 なら
%>
<tr bgcolor="#FF0000" align="left"><td><% = i %></td></tr>
<%
        ElseIf a = 1 Then 'a が 1 なら
%>
<tr bgcolor="#00FF00" align="center"><td><% = i %></td></tr>
<%
        Else              'a がそれ以外なら
%>
<tr bgcolor="#0000FF" align="right"><td><% = i %></td></tr>
<%
        End If            ' 条件分岐終了
    Next                  ' 次の i へ
%>
</table>
</body>
</html>
```

注：実行しても見にくいです（ぉ

言い忘れていましたが、別に一組の「<%」と「%>」の間だけでプログラムが完結している必要はありません。全体で完結していればいいのです。「<%」と「%>」の間以外の部分を Response.Write

で置き換えてみるとわかりやすいかもしれません。あと、「'」（カンマ）の後はコメントになります。

このプログラムで何をしているかというと、行が 9 つある表を作って、n 番目の行に n を 3 で割ったときの余りに応じたスタイル（背景色、テキスト配置）を割り当てています。よくわからないという人は、頭の中で i を増やしたりしてみましょう。そしてそのとき a がどうなってどの処理をして、と考えていくのです。簡単なループと簡単な条件分岐だけなのですぐにわかるようになりますよ。

終わりに

結構長くなってしまいましたね。9 ページにもなってしまいました。実を言うとブラウザから送られてきたデータを読み取って処理したりとか、そういったもうちょっと踏み込んだあたりのことも書きたかったんですが、私の体力がもう限界なのと（弱）締め切りをとっくに過ぎている（ぉ）[8]ことから、次回にまわしたいと思います。今回は本当に基礎の部分しかできませんでした。しかし、基礎というのはえてして重要なものなので、今回の内容はちゃんと理解しておきましょう。そうすれば次回以降楽ですよ :-) 次回のテーマは「フォームデータを受け取る」（予）です。こうご期待。

何か質問などのある方は kit@tsg.ne.jp まで。新歓コンパにも行くのでそこでとっ捕まえてもよしです。

最後に。ASP は身に付けると便利ですよ。何せバイトに使える！（ぉ

[8]現在朝の 5 時。眠い。

束めた ADSL と dyndns.org によるドメインの運用について

Tossy-2

はぢめに：姉さん、ADSL です。（誰や）

というわけで、「◯テル[1]」…ではなくて、ADSL です。

昨年から、ようやっと日本でも低価格の高速常時接続回線が手にはいるようになりました。ちょっとの間に東京・名古屋・大阪めたりっくをはじめとしてフレッツ ADSL やら eAccess やら J-DSL など、どんどん出てきてはエリア拡大を繰り広げてきました。

今まで T1 並の速度の回線と言ったら、今までは個人で持つにはちょっと高すぎ、せいぜい一般庶民としては、常時接続だったら OCN エコノミーあたりだったものですが、今では既に ADSL 主流に移行しているとも言えます[2]。

「常時」という点ではフレッツ ISDN も同様なのですが、やはり通信速度が段違いなので、現在の値段だったら明らかに ADSL の方が得です[3]。ただ、ADSL の不利な点は、新しいサービスのためにまだ全国規模では拡がっていないという点、また光収容[4]の回線では使えないという点ですか。NTT のページを見ると、どんどんエリア拡大は続いているようですが。

まあそんなわけで、うちでも ADSL を導入しました。昨年 11 月末に東京めたりっく通信に申し込み、今年 2 月下旬に開通。結局 3 ヶ月ほどかかりましたが、まあ特に開通に当たって何の障害もなく、スムーズに行きました。ちなみに、コースは Single（3 月末で申し込み終了しちゃいました。なんでやねーん！）です。

もうすぐ開通から 3 ヶ月が経とうとしていますが、その間にルータマシンの導入やら何やら、いろいろ有ったものです。覚え書き＋編集長への貸し作り兼莫大原稿で泣かす（ぉ）という意味も含め、動的 IP でドメインを運用（一応試験中ということで^^;）する顛末をつらつらつらつらと書いてみたいと思います。一応新歓号なので、まだ TSG 内の用語に慣れていない方も読むでしょうから、脚注をたくさんつけました[5]。

なんかでんぱっぱ〜♪ な文章（しかも長文だ！）になっちゃってるように見えなくもないのですが、まあだいたい人生なんてそんなもんです（わかめ）。諦めましょう。

なお、ここからしばらくはライトな文体が続きますが、途中からいきなりまじめ〜になるやも知れません[6]。ご注意をば。

---

[1] わっかるかなぁ？わっかんねぇだろうな…、って、これもネタじゃんか＞自分

[2] なお、ここでは usen のことは考えないことにします:D。

[3] まあ、この辺は個人の考え方でもありますが…少なくとも私はそう思いますね。

[4] 光ファイバーの経路が局から宅までの回線に入っていること。

[5] そうそう、つけすぎなぐらい（笑）。— え？いつもの癖だろうって？ はて、なんのことやら。:)

[6] こういう現象を相転移と言います。（嘘

自宅サーバは漢の夢

さて、まあ ADSL が開通してからしばらくは現在の（現在「でも」）メインマシンの Athlon750 自作機で Web 見たり大きなファイル（たとえば、FreeBSD の iso イメージなど…）をダウンロードしたりしていました。

速度も平均して、下りが 1.8〜1.9Mbps・上りが 220〜250kbps 程度です。なんかサービスの上限値超えてるようにも見えますが、まあ速い分にはいいです。気にしません。（ぉ

しかし、ひととおり速度を堪能すると、次に来るのは常時接続に対して沸きあがってくる歓喜です。今までのように、テレホタイム[7]に煩わされる事もない、いつでも気が向いたときにパソコン[8]の前に座れば接続中。おまけに回線帯域も持て余し気味。こうなったらやはり、「自宅サーバだ！」となってしまうのも世の常です[9]。

更に言えば、うちは Single 契約なので、サーバでもなんでも好きに立て放題です（i.e. 外からもアクセスし放題 XP [10]）。

こりゃあサーバ立てなあかんでしょう（笑　と一気にやる気が高まったのでした。

さて、ここのところでちょっと補足（興味なかったら読み飛ばしましょう）。なぜ私が Single を選んだのかと言うと、東京めたりっくの場合、ADSL サービス（当時）には Single と Family というメニューがあり、Single はブリッジタイプ、Family はルータタイプのモデムが来ることになっていました。

めたりっくは「1 台でつなぐなら Single、複数台をインターネットに接続するなら Family」と言っていましたが、ルータタイプのモデムには確かに、PC それぞれに PPPoE クライアントをインストールせずともよいという利点はあるものの、パスワードがかかっているのでポートフォワーディング等の設定をいじることができず、どうも融通が利かないようです[11]。

私は、作曲系のサークル（学校の、ではないですが）に所属しているので[12]（ちなみに `http://www.nerv.to/lunar/` あたりらしいです）、IRC やら ICQ でファイルをやりとりすることがしばしばです。こういうときに、ルータが間に挟まってるとちょっと厄介で、NAT[13]抜けをしなくてはならないので、ポートフォワーディングを設定しなければなりません。しかし、Family だとそういう部分がどうもできるのやらできないのやら不安が残ります。そこで、Single を選んだわけです。

以上ちょっと長々書きましたが、要はサーバ公開ができるということでした。そこでい

[7] 23:00〜翌 08:00 のこと。テレホーダイのサービスで（指定番号への）電話がかけ放題になる時間帯。
[8] 「ぱそこん」ニ非ズ（わかめ
[9] そうです。そういうもんなんです。私は直観力 ™ により（以下略
[10] 念のため。この XP は、某げいちゅ様の OS とはなんの関係もありません。ただの顔文字です。:D
[11] 先日、ようやっと設定ツールが配布開始されました。
[12] 他に理由なんてありませんよ？　たとえば、あんなものやこんなものをやりとりするとか。…って、何言わせようとしてますか！(;;)
[13] Network Address Translator、外へ出ていくパケットや中へ入っていくパケットの発信元や送信先アドレスを変換してくれる機構。

ろいろすべく、メインマシンに入れてあった FreeBSD をあれこれいじくりました。サーバ構築の話はし始めるとホントにキリが無くなるので省きますが、参考書などもいろいろありますので、そちらを参照されるとよいでしょう。

それから一ヶ月ほどはメインマシン一台での運用でしたが、サーバは FreeBSD、しかし音楽の作業などは Windows でやらないとならないので、だんだんと OS 変えるのが面倒になってきます。そこで秋葉原にて中古のメーカー製省スペースの PC を一台購入し、ルータマシン兼サーバとすべく設定[14]。

こうなるともう、ほとんど接続を切ることもなくほとんどつなぎっぱなし状態。たまに局側の反応が無くなることがあり、そんなときに一旦切断してから再接続をするぐらいで、ルータマシンもモデムも、ほぼ不眠不休で健気に頑張っております。今日もルータマシンの起動からの経過時間は増え続けていることでしょう。

dyndns.org といふもの

ここらへんからそろそろまじめに書きます（笑）[15]。

ADSL は開通した、サーバ（のプログラム）も準備した。あとは？——そう、ホスト名が必要ですね。

フレッツ ADSL や eAccess だとプロバイダを選択でき、その中には固定で IP アドレスを割り当ててくれるところもあるのですが[16]、東京めたりっくの ADSL は残念ながら動的 IP アドレスです[17]。つまり、再接続したときには、もらえる IP アドレスは変わる可能性が大きいということです。

そうなってくると、一番簡単なサーバ公開の手法は、毎回毎回アクセスする人に IP アドレスを知らせるか、自分でどこかに IP アドレスを書き込むかすることですが[18]、やはりそうでなく何か決まった名前で公開する方がいいと思います。覚えやすいですし、管理者側・ユーザー側両者ともに労力を節約出来ます。

そこで、こんなサービスがあります。

- http://www.dyndns.org/
- http://www.ns1.net/

見ての通り、好きなホスト名（ドメイン部分はいくつかから選べます）とＩＰアドレスを登録しておくと、DNS レコードを提供してくれるサービス（基本的には無料、寄付歓迎とのこ

[14]おかげで、メインマシンは現在ほぼ Windows マシンとなってしまっています。
[15]—— ということは、今まではまぢめに書いてなかったのデスカ？＞自分
[16]激遅とウワサの VC-NET や、InfoSphere などが一例です。
[17]ちぇっ。（謎
[18]なんかア●グラっぽ〜い（笑

と[19]）です。自動更新用のスクリプトなどもあるので、接続した後自動でデータベースを更新するように設定することも可能です。私は上の dyndns.org の方を選びました。

dyndns.org 初級：まずは使ってみませう。

使用にはまずユーザー登録が必要です。「Members NIC」から「New Account」を選び会員規約に同意すると、ユーザー名とメールアドレスを訊いてきますので、好きな名前を入れましょう。メールアドレスは確認のため 2 回入力する方式になっています。

フォームを送信すると、エラー無く登録された場合は仮パスワードが送られてきます。メールの中にログイン用の URL があるので、48 時間以内にユーザー名と仮パスワードでログインすれば、本登録が行われます。ここで実際に使うパスワードなどをまず設定します。
次に、Dynamic DNS のサービスに移ります（右のメニューに「Logged in as hogehoge」のように表示されていることを確認してください)。ホストが登録されていれば、

```
Hostname      Last Updated                              IP in Database
e.dyndns.org  Sat May 12 03:03:57 2001 EDT (-0400)  2.71.82.81
```

などと表示されますが、まだ登録したてでは何も表示されないはずです。

では、次にホストを登録しましょう。右のメニューから「Create New Host」を選び、ホスト名と IP アドレス（アクセス元の IP アドレスが自動的に書き出されます。Proxy とか通してるとそっちが出る恐れがありますので[20]、ちゃんと自分のアドレスであることを確認しましょう）を設定して「Add Host」ボタンを押します。名前の重複するホストがなければ、これで登録されるはずです。

IP アドレスを変更する場合は、ログイン（「Members NIC」から）した後「Dynamic DNS」から変更したいホスト名を選び、書き換えるだけです。Web ブラウザからできるので、非常に簡単ですね。

なお、dyndns.org では、Dynamic DNS のサービスで 30 日間アドレスの変更がない場合は、設定されたホストが削除されます。寄付すると使用できる Static DNS のサービスでは、このようなことはありません[21]。

さて、登録されたからといって、すぐにアクセスできるようになるかというと、そうも行きません。DNS が世界に一つしか無かったら大丈夫でしょうが、実際には DNS はあちこちに存在します。そして、それらはだいたい今まで受け取った問い合わせに対して探した答えをしばらく保持しています[22]。そのため、DNS 情報を変更すると、それがあちこちに伝播するまでに

[19]なお dyndns.org では寄付を払うと Static DNS ができるようになります。説明は後程。
[20]HTTP_X_FORWARDER とか環境変数見てれば、ちゃんと自分のアドレスが出ることもありますが…ここはどうなんだろう。
[21]…と書いてあります。
[22]キャッシュ(cache) といいます。

しばらく時間がかかるのです。

固定 IP アドレスの場合は、IP アドレスの変更はだいたい上流ネットワークの変更などを示します。このようなことはあまりないので、それほど気にする必要はないのですが、特に動的割り当ての IP アドレスの場合は、繋ぎ直す毎にほぼ毎回[23]変わりますので、固定に比べて遙かに変更の頻度が高いわけです。そのため、更新直後は（ネームサーバとして dyndns.org のものを使用していない限りは）古い IP アドレスにアクセスしに行ってしまうという事故が起きる可能性があります。

ということで、それなりにアクセスの有るサーバならば（たとえばその上でビジネスを行うなど）、やはりそれなりのサービスにし、固定アドレスを貰うべきでしょう。遊びあるいは研究のために立てるならば、動的割り当てでも構わないと思いますが、上記のような事故が起こりうることは、少なくとも知っておくべきだと思います。

dyndns.org 中級： オプションを見る

とりあえず、ホスト情報をみると、ホスト名と IP アドレスの他にもいくつかオプションが見えます。それぞれ、以下のような意味があります。

`Enable Wildcard`

ワイルドカードが有効になります。

たとえばホスト名を「hoge.dyndns.org[24]」とすると、普通は（チェックをつけなかった場合）「www.hoge.dyndns.org」のような名前は解決できませんが、Enable Wildcard にチェックをかけると、そのようなホスト名も使えるようになります。

もう少し正確には、

```
$ORIGIN         dyndns.org.

pi      IN      A       3.141.59.26
*.pi    IN      CNAME   pi
```

という感じの設定がなされるわけです[25]。ここで * は、任意の文字列に対応します。

`Mail Exchanger`

ネームサーバの MX[26] レコードに指定するホストを書きます。

自分でメールサーバを設定している場合には、ここにホスト名を書けば、

[23] 運が良ければ、同じ IP アドレスが割り当てられることもありますけどね。

[24] ちなみにこれは、某 TSGer によりすでに登録されています（@ 05/15 現在）。

[25] 詳細はオライリーの「DNS&BIND」（バッタ本）を参照してください。現在第３版が出ています。

[26] **M**ail e**X**changer の略。

「myname@hoge.dyndns.org」などでメールを送れるようになります[27]。

`Backup MX`

Mail Exchanger 設定と組み合わせて使います。チェックすると、「Mail Exchanger」に設定されたホストはセカンダリのメールサーバ（プライマリサーバに送れないときに使われるバックアップサーバ）になります。この場合プライマリサーバは、設定しているホスト名（ここでは hoge.dyndns.org など）になります。

…と、一応一通り説明しましたが、まあ普通に使っている限りは、アドレスとホスト名の部分のみ気にしていればいいわけです。ハイ。

dyndns.org 上級： クライアントを使ってみる

dyndns.org には、Web の他にクライアントプログラムでデータベースを更新する方法もあります。「Dynamic DNS」のページからメニューの「Clients」→ OS を選択すると、リストが表示されます。

いろいろ種類がありますが、FreeBSD などの UNIX 系の場合は、ipcheck.py がよいと思います。dyndns.org クライアントとしての要件を全て満たしており、Python スクリプトで記述されているのでインタプリタ[28]があればどの OS でも使えます。

とりあえず、`http://ipcheck.sourceforge.net/` からスクリプトをダウンロードできます。

Python がインストールされていることを確認の上、実行してみましょう。詳しい使い方は、Web ページに説明されています。オプションをつけずに実行するとヘルプが表示されます（だいたいよく使う部分を抜粋）。

```
# ./ipcheck.py
Usage  : ipcheck.py [options] Username Password Hostnames
or       ipcheck.py [options] --acctfile acct_info_file
...
```

アップデートの仕方は、まあ使用例の通りにすれば問題ありません。ユーザー名やパスワードを見られる可能性があるのでコマンドラインに書くのはいやだ、という場合は、
texttt–acctfile オプションを使い、

```
muha password hoge.dyndns.org
```

[27] hoge.dyndns.org 宛のメールを受け取るように設定しなければなりません。たとえば、CF を使って sendmail.cf を作成しているなら、`MY_ALIAS` や `ACCEPT_ADDRS` に書き加えればいいはずです。
[28] パッケージとか探してみると多分あるはずです。

のように記述したアカウント情報のファイルを指定してやれば OK です。
また、FreeBSD の場合は、ppp が使用するインターフェースの名前が tun なんたらなので、-i オプションを指定してやる必要があります。あるいは、スクリプトを書き換えてもよいです。

```
opt_interface = "ppp0"
```

という行があるはずなので、ppp0 を tun0 に書き換えてやります。
さて、では実際に使ってみます。以前のデータベースアップデートから IP アドレスが変っていることを確認して、

```
# ./ipcheck.py --acctfile ipcheck.acct
```

のように実行します。うまくアップデートされた場合は、

```
ipcheck.py: pi.dyndns.org good 3.141.59.26 -update successful
```

などと出力されるはずです。
また、IP アドレスが変更されていないにも関わらず、何度も更新しようとすると、abusive と表示されます（通常は ipcheck の側で IP アドレスが変更されているかどうかチェックしてくれます。-f オプションを指定しない場合、変更が無い場合はアップデートしにいきません。）。abusive なリクエストを何度か行うと、アカウントが停止されます。注意してください。

このように、クライアントを使えば、接続から dyndns の設定までを一括処理させることも可能です。[29][30]
実際の例は、次節にて。

不安定回線と戦うの法

東京めたりっくの ADSL は、回線速度が ADSL サービスの中ではかなり速い部類にはいるのですが、実のところ、回線が（比較的？）不安定なのが玉に瑕です。
常時接続だと、自宅でマシン動かしたままほっといて外出したり…なんてことは日常茶飯事なのですが、弱ったことに、たまーに回線の反応が無くなってしまうことがあります[31]。そういうときは再接続してやればよいだけですし、接続に使っている iij-ppp にも、回線が切断されたら自動的に再接続を行う機能は付いているのですが、この現象は「回線切断」ではないので、iij-ppp の自動再接続は働かないのです。
ちなみに実話ですが、しばらく前の情報棟分科会の日のこと。17:30 頃には確かに回線が生きていたのに、18:00 過ぎになったら何故かつながらず、Italk 講習会ができなかった…とかいうことが[32]。

[29] /etc/ppp/ppp.linkup に記述するというのは残念ながら無理のようでした。
[30] ていうか、ppp でコマンドラインを起動すると終了待ってくれるのね、律儀に。…まあ当然か。
[31] 今までに 2 回ほど経験しました。2 週間ぐらいずーっとつないでたら、いつのまにやら反応が消えてて…(汗
[32] がんばって、情報棟内から繋げるようにゲートウェイ作ったのに(T-T)

それと戦うためには、定期的に外部に ping を打ち、反応が無くなった場合は再接続処理を自動で行うようにスクリプトを書けばよいわけです。じゃあ、ちょこちょこと書いてみましょうか。ホントはシェルスクリプトかなんかで書くとよさげなんですが（軽いし？）、私は残念ながらよく知らないので、へっぽこに Perl でげしげしと書きます。[33]

```
#!/usr/bin/perl

#変数宣言
$ppp_pid_dir = '/var/run';              # *.pid のあるディレクトリ

sub killppp {
    # tun を指定して ppp を殺す (SIGTERM)。
    local($tunname) = @_;
    local($ppp_pid);

    $ppp_pid = "$ppp_pid_dir/$tunname" . '.pid';
    if(-e $ppp_pid){
        open(FILE, "$ppp_pid") || die;
        $pid = <FILE>;
        close(FILE);

        if($pid > 10){
            kill(15, $pid);
            print "killed ppp on $tunname\n";
        }
    }
}

sub searchtun {
    # 動いている tun(=ppp) を探す。
    local(@result, @data);

    opendir(DIREC, $ppp_pid_dir) || die;
    @data = readdir(DIREC);
    close(DIREC);

    foreach $_ (@data){
        if(/(tun\d+)\.pid/i){ # tun*.pid を探す。
            push(@result, $1);
        }
    }
    @result;
}

sub killall_ppp {
```

[33]なお、何も考えずぼけーっと組んでたので、あんまりスマートじゃないです。

```
        # 全ての ppp を殺す。
        local(@tunlist) = &searchtun;
        foreach $_ (@tunlist) {
            &killppp($_);
        }
    }

     sub islinealive {
        # 回線が生きているかチェック。
        # ・コマンドラインは適当にいじくること。
        local($recvp) = 0;
        open(PIPE, "/sbin/ping -c 1 -t 3 xxx.xxx.xxx.xxx |") || die;
        while(<PIPE>){
            if(/(\d+) packets received/i){
                print;
                $recvp = $1;
            }
        }
        close(PIPE); # ひどく後ろ向きな気分だ。
        $recvp;
    }

     sub connect {
        # 残っている ppp を殺し、再接続する。
        # ・ppp のオプション設定は適切に書き換えること
        &killall_ppp;
        system('/usr/sbin/ppp -quiet -dedicated -nat <hogehoge>');
        sleep(5);   # 5 秒待つ
        ($tun) = &searchtun;
        if($tun eq ''){ die; }
        print "ppp started on $tun\n";

        # dyndns.org のデータベースを更新する。
        # ・コマンドラインは適当に書き換えること。
        system("( cd /root/dyndns ;" .
               " ./ipcheck.py --acctfile ipcheck.acct " .
               "-i $tun <hostnames> )");
        print "IP updated\n";
    }

    unless(&islinealive){
        &connect;
    }
```

とりあえずこんな感じでしょうか。なお、このスクリプトは FreeBSD 用なので、Linux などではそのままでは動かないかも知れません。

このスクリプトを cron に仕掛けるなどして定期的に（うちの場合は毎時 10,40 分に実行するように設定してます）実行すれば、反応が無くなったときに再接続して dyndns.org のデータをアップデートしてくれます。

独自ドメイン、ゲットだぜ ～汎用 JP ドメインのバヤイ～

最近 ZDNN[34] あたりを見ると、時折「汎用 JP ドメイン」なる名称がトピックに現れています。

今まで、.jp ドメインは、第２レベルドメインとして組織種別を表すドメインがくっついていました。.“***or***”.jp とか、.“***ne***”.jp とかいうやつです。しかし JPNIC の方針転換・規制緩和[35]により、第２レベルの開放の他にも、日本語ドメイン名を含めたドメイン名なども登録できるようになりました。これが汎用 JP ドメインというわけです。詳細は、JPRS（Japan Registry Service Co.,Ltd.）のサイト `http://jprs.jp/` にあります。

さて、ドメインを取るにはいくらかのお金がかかります。また、ドメインを運用していく上で維持費を NIC に払わなくてはなりません。しかし、ドメインの維持費や登録費用（だいたい初年度分の維持費＋手数料αのようです）はそれほど高いわけではありません。登録業者にもよりますが、たとえば .to の場合は年間 $50、.com や .org の場合は、だいたい年間 $25～$35（Network Solutions[36]の場合）ぐらいです。

ちなみに汎用 JP ドメインの維持費はというと、JPRS に直接申請した場合で、年間 7,000 円（初期費用は、初年度維持費を含んで 14,000 円）。他の事業者の登録代行サービスを利用すれば、たとえば私の利用したヒューメイア[37]の場合は、年間 3,500 円（初期費用は、初年度維持費に加えて手数料と利用者登録の費用が 3,000 円ほどかかります）です。年間 3,500 円ということは、月々 300 円弱程度なので、（それなりに安いところを選んで、数個登録する限りは）あまり経済的負担になるということはありません。

また、汎用 JP ドメインには接続義務がないので、ドメインを取るだけ取っておいても構いません（ちなみに、5 月 7 日から先願申請に切り替わりましたので、現在では、登録されていないドメインは早い者勝ちで登録されます）。

おそらく将来的にはインターネットは専用線が主流になる…と思いますので、その時のために取っておくというのも一興かと。

実際の取得の手順については、それぞれの事業者間で異なっていますので、登録サービスのページを見ましょう。なお、サービスやってる事業者がかなり沢山ありますので、どこがいいかはあちこち比べて選ぶとよいと思います[38]。

---

[34] `http://www.zdnet.co.jp/news/`
[35] 接続義務が無くなった、多言語ドメイン対応、など。
[36] `http://www.networksolutions.com/`
[37] `http://www.humeia.ad.jp/`
[38] 私は、「ここいいよー」と言われて、それでヒューメイアに決定しちゃいましたけどね。

運用開始ノススメ

ドメインを取ったら、まあそのまま遊ばせておいてもいいのですが、やっぱり常時接続の環境があることだし、運用してみたくなりました[39]。ここでは dyndns.org と組み合わせて運用する方法を紹介します[40]。

ドメインを実際に使うには、DNS が必要です。まあ、当たり前ですね。
もちろん、誰か知り合いで固定アドレスをもらっている人に DNS を引き受けてもらうという手もありますが[41]、ここでは自分で DNS を用意することにします。その方が、ゾーン情報をいじくったりするのが簡単にできます。
しかし、うちは何度も言うように[42]動的割り当ての IP アドレスなので、ちょっと問題があります。

NIC データベースに登録するホストをどうするか。

ドメインを使う場合は、DNS を NIC のデータベースに登録してもらいますが、これがかなり時間を食います。
実際、私が登録しようとしたときは、処理受付から実際に登録されるまで丸一日近くかかりました[43]。なので、データベースはそう頻繁に更新するわけにもいきません。

ゾーン情報について。

資源レコードには、A[44] やら MX やらいろいろありますが、動的 IP アドレスだと、A レコードを IP アドレスが変わったときに毎回更新してやらなければなりません。

ここで 2 番目の問題はまさしく dyndns.org を使う意義です。また 1 番目の問題も、NS レコードには FQDN[45] を記述できるので、dyndns.org で使っているホスト名を使って問題ありません。
ということで、ネームサーバを to2.ath.cx で登録し、named のゾーンファイルを以下のように書きました[46]。

```
$TTL    3600
```

[39] なりますよね？　ね？(ぉ)
[40] いよいよクライマックス。○戸黄門で言えば終了前５分ぐらいですか。
[41] TSGer にもいますし、それ以外でも、たとえば私の場合は nerv.to の管理を引き受けてもらっている人がいます。
[42] くどいです（笑）
[43] 毎日データベースを更新する時間があって、それをちょっと過ぎたところで申請したのか、あるいは…。
[44] Address の略。
[45] Fully Qualified Domain Name。完全なドメイン名。たとえば情報棟からだと、WWW 実験のサーバは user だけで参照できますが、これに対し FQDN は、user.ecc.u-tokyo.ac.jp です。
[46] ホスト名やらドメイン名は気にしないでください。…というか、このゾーン定義は実物ですけどネ！(ぉ)

```
@       IN      SOA     eva-01.jp. root.eva-01.jp.  (
                               1       ; Serial
                               3600    ; Refresh
                               900     ; Retry
                               3600000 ; Expire
                               3600 )  ; Minimum
        IN      NS      to2.ath.cx.
        IN      MX      10      to2.ath.cx.

magi    IN      CNAME   to2.ath.cx.

mail    IN      CNAME   magi
www     IN      CNAME   magi
```

ちなみに、見ての通り、A レコードがありません。その辺の解決は、dyndns.org に丸投げしました（爆）[47]。

また MX レコードの右辺には `mail` と書きたいところですが、一応お約束として A レコードの左辺となっているものを記述しなければならないので、このようになっています。

さて、これでちゃんと実際に外から見えるかどうか…。

答えは、`http://www.eva-01.jp/` をブラウザで開いてみれば分かります（やはり CNAME ばかりなので、アドレスが引けるまでちょっと時間がかかってしまうようですが[48]…）。まあそんなわけで、冒頭のメールアドレスにも、自宅のアドレスを載せてみました。

では、最後に

えー、まずはここまで読んでくれた方に贈る言葉がありますデス。

お疲れ様でした。

…というか、僕自身もこんなに長文書くはめになるとは思いませんでした[49]。部報は結局何ページになるのかまだわかりませんが、この原稿がかなり圧迫してるような気がしてなりません。あーあ。やっちまった。しかし、自宅サーバは何となーく夢だったので、書いてるうちについついこんな量になってしまいました。

まあ、いつも通りの一人ボケツッコミのでんぱっぱ文章ですが、2 日でがしがし書き上げた原稿なので、赦してください[50]。

[47] おおあなたひどいひと。私に（以下略）＞自分

[48] A レコード自分持ちにしようかとも思ったんですけどねー…。更新する場所が 2 カ所あると、やはりめんどくさいですし。とりあえず一番楽な方法を選びました。

[49] …って、自分から書き出したんじゃないか（爆）。

[50] 言い訳になっとらんぞ＞自分

では、内容について。

一応新歓号なのでそれなりに解説はつけたつもりでいます。ネットワークは奥がかなり深いので、サーバ構築などに関してはやはり専門書を別に読む必要があるかと思いますが。あれこれ調べて書きましたが、記述の不正確な部分等ありましたら筆者までどうぞ。

この文章が、「せっかくだから俺はこの赤の扉[51]、もといADSLを引いて、サーバ立てるぜ！」という人の参考になれば幸いです。

…って、まぢめにまとめすぎたかナ？

DirectDraw の半透明処理第一回

まめ

はじめに

まめです。今回はDirectDrawでサーフェスのロックを使った16bit色での半透明処理について書いてみようと思います。この話は16bit色限定です。全三回で書く予定にされました。

半透明処理とは

DirectDrawで通常の矩形転送（IDirectDrawSurface7::Blt）をすると、当たり前ですが描画をした部分の背景は完全に見えなくなります。半透明処理は、背景と描画するスプライトの色をまぜて表示しようとするものです。原理は簡単です。背景部分の色を色の三原色（R：赤、G：緑、B：青）で分けます。それを

{R，G，B}

とします。そこに表示するスプライトの色も同様に

{r，g，b}

と分けます。そしてそこに

$$\{R \times \alpha + r \times (1 - \alpha), G \times \alpha + g \times (1 - \alpha), B \times \alpha + b \times (1 - \alpha)\}$$

（ただしαは $0 \leq \alpha \leq 1$ なる実数）

[51]ECOLE のページ、いつのまにか熱いですね。

の色を表示します。この処理をスプライトのすべての点において行ないます。

α が 0 に近いほど背景色に近くなり、薄い表示になります。逆に α が 1 に近いほど背景色がかき消された濃い表示になります。

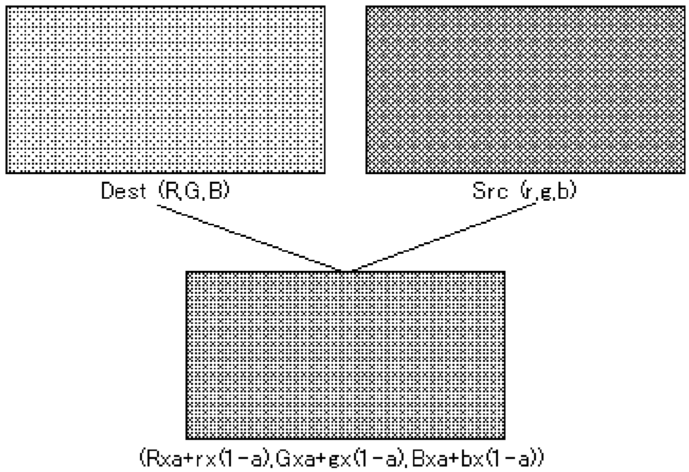


図 4.1: 半透明の原理

オフスクリーンサーフェス

では、具体的に DirectDraw でどのように行なうかです。

DirectDraw で扱うサーフェスとは、言ってみれば絵を描く Canvas[1] のようなものです。

サーフェスには一つだけ特別なサーフェスがあります。プライマリサーフェスです。プライマリサーフェスは画面に表示されるサーフェスです。プライマリサーフェスでないサーフェスはオフスクリーンサーフェスです。一般的にはオフスクリーンサーフェスにスプライトを置き、そこからプライマリサーフェスへ矩形転送して表示します。一般的にはプライマリサーフェスの前にバックサーフェスというサーフェスを用意し、オフスクリーンサーフェスからバックサーフェスに転送し、必要な転送（描画）がすべて終わったらバックサーフェスからプライマリサーフェスに一度に転送します。

ここで、オフスクリーンサーフェスには 2 種類あります。そのデータをビデオメモリ上に置くオフスクリーンサーフェスと、システムメモリ上に置くオフスクリーンサーフェスです（プライマリサーフェスは常にビデオメモリ上にあります）。

[1]英語が嫌いな私があえて Canvas と記述したのに意味は…ないことにします。

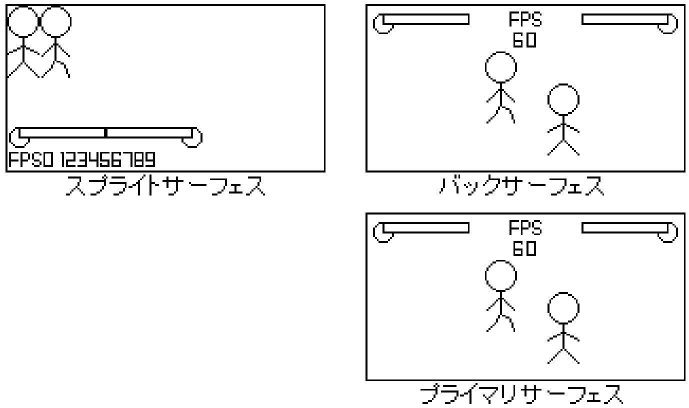


図 4.2: DirectDraw の一般的なサーフェスの扱い

この違いは、転送スピードです。ビデオメモリ間の転送（たとえばビデオメモリ上のオフスクリーンサーフェスからプライマリサーフェスへの転送）はビデオカードが行なうため一般に高速です。ビデオメモリとシステムメモリを行き来する転送（たとえばシステムメモリ上からプライマリサーフェスへの転送）は比較的低速です。

ならばビデオメモリ上のオフスクリーンサーフェスの方が絶対的に得に聞こえますが、そうでもありません。一般的にビデオメモリはシステムメモリに比べ量が少ないですから、仕方なくシステムメモリ上にオフスクリーンサーフェスを取る場合もありますし、意図的にシステムメモリ上にオフスクリーンサーフェスを取る場合もあります。それがサーフェスをロックする場合です。サーフェスのロックとは、サーフェスのメモリに直接アクセスして読みとったり書き込んだりすることです。メモリのデータを直接 CPU で読み書きするならビデオメモリよりシステムメモリの方が高速に処理できます。ということで、半透明処理などのようにサーフェスをロックして直接読み書きする処理ならばサーフェスはシステムメモリ上に取った方がいいです。

つまり、半透明処理などエフェクトをかける転送の場合はシステムメモリ間で、その他の転送はビデオメモリ間で行なうのがよいということになります。

以下の図は 2 種類のオフスクリーンサーフェスをともに使用する場合の図です。

サーフェスのロック

サーフェスをロックしてメモリを直接読む場合、データはどのような形式でメモリにあるのかを調べる関数が `IDirectDrawSurface7::GetPixelFormat` です。このデータ形式はビ

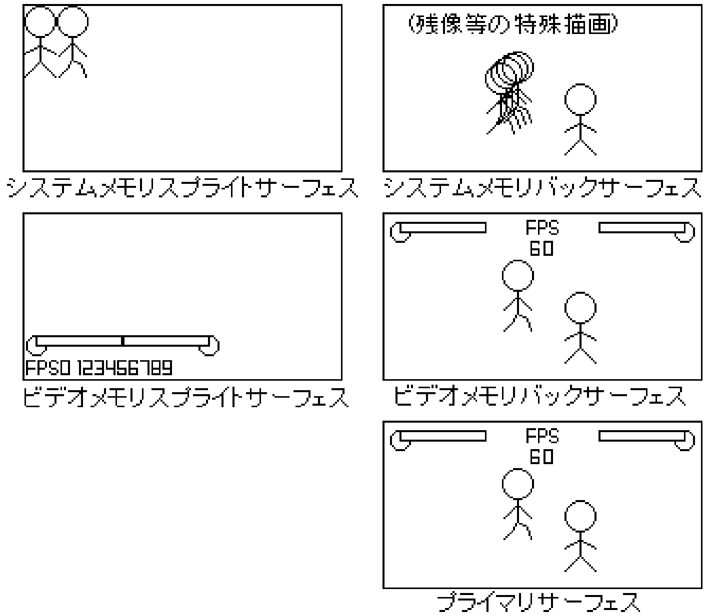


図 4.3: 2 種類のオフスクリーンサーフェスを考慮したサーフェスの扱い

デオカードに依存してしまうので、サーフェスのロックはこのデータ形式別の処理を行なうか、すべての場合を処理できるようにコードを書かなければなりません。ここが最高にうざったい部分です。デバッグには各種ビデオカードでテストしないといけないことになります。なんで規格を統一しなかったのか疑問です。一般的な形式は以下の 3 種類のようです。

RGB-565 型

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

00 〜 04 ビットに B のデータを、05 〜 10 ビットに G のデータを、11 〜 15 ビットに R のデータを保存する 16 ビットです。

BGR-565 型

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
B	B	B	B	B	G	G	G	G	G	G	R	R	R	R	R

RGB-565 型の B の位置と R の位置が逆転したものです。

RGB-555 型

F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
0	R	R	R	R	R	G	G	G	G	G	B	B	B	B	B

15 ビット目は常に 0 です。

RGB-565 が最も多い形式のようです。うちのビデオカードも、305 のマシン[2]のビデオカードもこれです。G のビットが 6 であるのは、人間の目は緑を感知する能力が高いためらしいです。

50 % 半透明処理

実際に $\alpha = 50\,\%$ の透過処理[3]を含む半透明処理をやるコードを示します。Visual C++ 6.0 です。見にくいソースですいません。

ソース中の MaskRGB はビデオカード依存の変数です。具体的には以下の値です。

$$\mathrm{MaskRGB} = \begin{cases} \texttt{0111101111101111(2)} & \text{（RGB-565 または BGR-565 の場合）} \\ \texttt{0011110111101111(2)} & \text{（RGB-555 の場合）} \end{cases}$$

なお、(2) は 2 進数という意味です。

```
VOID PutBlend(int x,int y,RECT rcSrc)
{
  RECT    rcDest={x,y,x+rcSrc.right-rcSrc.left,y+rcSrc.bottom-rcSrc.top};
  static  DDSURFACEDESC2  ddsd1,ddsd2;
  static  LPWORD          data1,data2;
  static  long            AddPitch1,AddPitch2;
  static  UINT            Buff,Buff2;
  int                     i,j;

  // クリッピング（画面からはみ出る部分を削る）
  if(rcDest.left  <0  ) rcSrc.left  +=  -rcDest.left  ,rcDest.left  =0;
  if(rcDest.top   <0  ) rcSrc.top   +=  -rcDest.top   ,rcDest.top   =0;
  if(rcDest.right >=sx) rcSrc.right +=sx-rcDest.right ,rcDest.right =sx;
  if(rcDest.bottom>=sy) rcSrc.bottom+=sy-rcDest.bottom,rcDest.bottom=sy;

  i=rcSrc.right-rcSrc.left;
  j=rcSrc.bottom-rcSrc.top;

  if(i>0 && j>0)
  {
    ZeroMemory(&ddsd1,sizeof(DDSURFACEDESC));
    ZeroMemory(&ddsd2,sizeof(DDSURFACEDESC));
    ddsd1.dwSize=ddsd2.dwSize=sizeof(DDSURFACEDESC);

    // スプライトサーフェスのロック
    if(lpDDSSystemSprite->Lock(NULL,&ddsd1,DDLOCK_WAIT,NULL)!=DD_OK) return;
    // バックサーフェスのロック
    if(lpDDSSystemBack->Lock(NULL,&ddsd2,DDLOCK_WAIT,NULL)!=DD_OK)
    {
```

[2] Urd 機です。ほかの 305 マシンは未調査。
[3] スプライトのカラーが 0（黒）の部分は転送しない処理。

```
          lpDDSSystemSprite->Unlock(NULL);
          return;
        }

          AddPitch1=ddsd1.lPitch>>1;
          AddPitch2=ddsd2.lPitch>>1;

          data1=(LPWORD)ddsd1.lpSurface+rcSrc .left+rcSrc .top*AddPitch1;
          data2=(LPWORD)ddsd2.lpSurface+rcDest.left+rcDest.top*AddPitch2;

          Buff=MaskRGB;

          // メイン
          for (y=0;y<j;y++)
          {
            for (x=0;x<i;x++)
              ((Buff=data1[x])!=0x0000)
              // 透過処理（カラー 0 （黒）の部分は転送しない）
            {
              data2[x]=((data1[x]>>1)&Buff)+((data2[x]>>1)&Buff);
              // 半透明処理
            }
          data1+=AddPitch1;
          data2+=AddPitch2;
        }
        // サーフェスのアンロック
        lpDDSSystemSprite->Unlock(NULL);
        lpDDSSystemBack->Unlock(NULL);
      }
      return;
    }
```

lPitch は、次のライン開始までのバイト単位の距離です。半透明処理の部分は以下のようになっています。

RGB-565 の場合（BGR-565 も同様）

```
data1[x]          =r1r2r3r4r5g1g2g3g4g5g6b1b2b3b4b5(2)
data1[x]>>1       =0 r1r2r3r4r5g1g2g3g4g5g6b1b2b3b4(2)
Buff              =0 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 (2)
(data1[x]>>1&Buff)=0 r1r2r3r40 g1g2g3g4g50 b1b2b3b4(2)

(data2[x]>>1&Buff)=0 R1R2R3R40 G1G2G3G4G50 B1B2B3B4(2)

結果=R’1R’2R’3R’4R’5G’1G’2G’3G’4G’5G’6B’1B’2B’3B’4B’5
```

RGB-555 の場合

```
data1[x]=0 r1r2r3r4r5g1g2g3g4g5b1b2b3b4b5(2)
data2[x]=0 R1R2R3R4R5G1G2G3G4G5B1B2B3B4B5(2)

(data1[x]>>1&Buff)=0 0 r1r2r3r40 g1g2g3g40 b1b2b3b4(2)
(data2[x]>>1&Buff)=0 0 R1R2R3R40 G1G2G3G40 B1B2B3B4(2)

結果=0 R’1R’2R’3R’4R’5G’1G’2G’3G’4G’5B’1B’2B’3B’4B’5
```

半透明処理の最大の課題はスピードです。ちなみにこの関数を 16x16 ドットの転送に 50000 回実行するのにかかった時間はおよそ 500ms[4] でした。つまり 1 回の実行に 0.01ms です。

うちのマシンの CPU は PentiumIII の 750MHz です。

RPG やシミュレーションなどの、あまりリアルタイムな動作を要求しないゲームならばこの程度のスピードでよいかもしれませんが、シューティングの弾幕に半透明処理を使用したり、アクションゲームで残像などに半透明処理を使用する場合は、60fps[5] を再現するためには 1 フレームの描画を 15ms に抑える必要があり、半透明処理以外の描画処理も必要であることを考えるとこれでは遅い可能性があります。

高速化

高速化と言えばアセンブラです（謎）。とりあえず 32bit で書き直して見ました。アセンブラなのにインデントされているのは私がアセンブラになれてないからです。気にしないでください（笑）。

```
VOID PutBlendAsm(int x,int y,RECT rcSrc)
{
    （中略）

    AddPitch1=ddsd1.lPitch;
    AddPitch2=ddsd2.lPitch;
    data1=(LPWORD)ddsd1.lpSurface+rcSrc .left+rcSrc .top*(AddPitch1>>1);
    data2=(LPWORD)ddsd2.lpSurface+rcDest.left+rcDest.top*(AddPitch2>>1);

    i>>=1;
    AddPitch1-=i;
    AddPitch2-=i;
    Buff=MaskRGB;

    // メイン
    _asm
    {
      mov     esi,data1       // 転送元
      mov     edi,data2       // 転送先
      mov     edx,Buff
      mov     eax,edx
      shl     edx,16
      or      edx,eax         // MaskRGB 32bit
      mov     eax,i
      shr     eax,1
      mov     i,eax
      mov     eax,j
_y_loopstart:
      mov     ebx,i
      push    eax
```

4 timeGetTime() 関数にて測定。timeBeginPeriod(1); にしてあります。
5 frame per second の略。

```
_x_loopstart:
      mov     eax,[esi]
      mov     ecx,[edi]
      shr     eax,1
      and     eax,edx
      shr     ecx,1
      and     ecx,edx
      add     eax,ecx
      mov     [edi],eax
      add     esi,4
      add     edi,4
      dec     ebx
      jnz     _x_loopstart
      pop     eax
      add     esi,AddPitch1
      add     edi,AddPitch2
      dec     eax
      jnz     _y_loopstart
    }
    // サーフェスのアンロック
    lpDDSSystemSprite->Unlock(NULL);
    lpDDSSystemBack->Unlock(NULL);
  }
  return;
}
```

相当いい加減に組んだので手抜き処理が目立ちます。透過処理していません。これで 50000 回の実行に 200ms です。約 2.5 倍のスピード アップ[6]です。

さらなるスピード アップを求めて、MMX を使って見ました。今度はクリッピングしてます。

```
VOID PutBlendMMX(int x,int y,RECT rcSrc)
{
  （中略）

    AddPitch1=ddsd1.lPitch;
    AddPitch2=ddsd2.lPitch;
    data1=(LPWORD)ddsd1.lpSurface+rcSrc .left+rcSrc .top*(AddPitch1>>1);
    data2=(LPWORD)ddsd2.lpSurface+rcDest.left+rcDest.top*(AddPitch2>>1);

    Buff=MaskRGB;
    _asm
    {
      mov     esi,data1       // 転送元
      mov     edi,data2       // 転送先
      mov     eax,Buff
      mov     ebx,eax
      shl     eax,16
      or      eax,ebx
      movd    mm4,eax
      movq    mm0,mm4
      psllq   mm0,32
      por     mm4,mm0         // 以上 RGB マスクを作る処理。
      mov     eax,j
      mov     ebx,i
```

[6]いい加減な処理なので本来の処理と同等の処理をさせた場合結構スピードが落ちると思います。

```
            mov     ecx,AddPitch1
            mov     edx,AddPitch2
            sal     ebx,3
            sub     ecx,ebx
            sub     edx,ebx
            sar     ebx,3
       _y_loopstart:
            mov     ebx,i
       _x_loopstart:
            movq    mm0,[esi]       // mm0 にスプライト 4 ドット分読込
            movq    mm1,[edi]       // mm1 に背景部分 4 ドット分読込
            pxor    mm3,mm3         // mm3 をクリア
            pcmpeqw mm3,mm0         // マスクを作成
            psrlw   mm0,1           // mm0 を 1 つ右シフト（2 で割る）
            pand    mm0,mm4         // mm0 を MaskRGB でマスクする
            movq    mm2,mm1         // mm1 を mm2 にコピー
            pand    mm1,mm3         // mm1 は 透過する部分のみ残す
            pandn   mm3,mm2         // mm3 は 透過しない部分のみ残す
            psrlw   mm3,1           // mm3 を 1 つ右シフト（2 で割る）
            pand    mm3,mm4         // mm3 を MaskRGB でマスクする
            por     mm1,mm3         // 透過する部分としない部分の論理和
            paddw   mm0,mm1         // スプライトと背景の和を取る
            movq    [edi],mm0       // 背景部分にスプライトを書き込む
            add     esi,8
            add     edi,8
       _x_loopend:
            dec     ebx
            jnz     _x_loopstart
            add     esi,ecx
            add     edi,edx
            dec     eax
            jnz     _y_loopstart
            emms
          }
          // サーフェスのアンロック
          lpDDSSystemSprite->Unlock(NULL);
          lpDDSSystemBack->Unlock(NULL);
        }
        return;
      }
```

今度は透過処理してます。これで 50000 回に 175ms です。あまり差はありませんでしたが、MMX なら pcmpeqw という透過処理に使いやすいニーモニックがあるので便利です。

MMX は 64bit を一度に扱うので 16bit 色では一度に 4 ドットを扱います。そのせいで実はこの関数は横幅は 4 ドット単位でしか扱えません。しかし、4 ドット以外の場合の処理に対応してスピードを落とすよりは、関数を実行する際に 4 ドット単位で指定すればいいと考えてこのままにしました[7]。

MMX 部分の流れを説明します。

```
>movq      mm0,[esi]
>movq      mm1,[edi]
```

[7]めんどくさいと言うのが本音であることは自明です。（ぉ

ここは単純にスプライトと背景を MMX レジスタに読み込みます。MMX レジスタは 64bit だから、1 ピクセル 16bit のドットを 4 つ読み込みます。

```
>pxor      mm3,mm3
>pcmpeqw   mm3,mm0
```

透過処理の核の部分です。pcmpeqw とは、64bit の MMX レジスタを 16bit ずつ 4 つのパックに割り、対応するパックとパックが等しい場合は第 1 オペランドに `1111111111111111(2)` を、等しくない場合は `0000000000000000(2)` を代入します。

例[8]

```
pcmpeqw mm0,mm1
   mm0 : 0000000000000000 0000000000000000 0000000000000111 0111000111000111
   mm1 : 0000000000000000 0000000000000001 0111000111000111 0111000111000111
 結果          真               偽               偽               真
   mm0 : 1111111111111111 0000000000000000 0000000000000000 1111111111111111
```

つまりこのニーモニックにより、透過する部分（カラーが `0000000000000000(2)` の部分）のビットのみをたてることが出来ます。

```
>psrlw     mm0,1
>pand      mm0,mm4
```

2 で割ります。ですが、`psrlw` は 64bit を 16bit ずつ 4 つのパックに割って右シフトします。つまり `0x0003 0000 0000 0000` を 1 つ右シフトしたとき `0x0001 8000 0000 0000` でなく、`0x0001 0000 0000 0000` となります。よって、各ドットについて 2 で割ったことになります（この場合次にマスクをかけるのでどちらでもよいのですが）。

```
>movq      mm2,mm1
```

背景部分をコピーします。

```
>pand      mm1,mm3
```

mm1 を透過する部分のみ残します。

```
>pandn     mm3,mm2
```

pandn は第 1 オペランドを反転させてから、第 2 オペランドとの論理和を第 1 オペランドに代入します。つまり、mm3 に透過しない部分のみを残します。

```
>psrlw     mm3,1
>pand      mm3,mm4
```

透過しない部分を 2 で割ります。

```
>por       mm1,mm3
```

背景部分の論理和を取ります。これでスプライトが表示される部分だけ 2 で割られた色 4 ドットが作成されます。

```
>paddw     mm0,mm1
```

スプライトと、スプライトが表示されるだけ 2 で割られた背景を足します。

```
>movq      [edi],mm0
```

---

[8] この例は Intel の”インテル・アーキテクチャ・ソフトウェア・ディベロッパーズ・マニュアル中巻：命令セット・リファレンス”から取ってきたんですが、mm1 の 2 つ目のパックの値を 0000000000000001 に変えてあります。ってゆうか Intel のほうが間違ってません？ どうみても 2 つ目が真なんですが…。

ちなみにこの関数を、メインの部分をコメントアウトして実行したとき、50000 回で 140ms[9] ほどでした。

以下次号。次号では「α 値任意の半透明処理」をやります。

酒との上手な付き合い方

植原 洋介 (ue@tkc.att.ne.jp)

新入生の皆さんはじめまして。97 年度に TSG の部長をしていたうえといいます。現在は大学院の修士 2 年です。[1]

さて、皆さんは大学に入ったわけで、当然酒を飲む機会が多くなると思います。酒は非常に人生を楽しくしてくれるものですが、多少の知恵を持って飲まないとあまり楽しくないし、さらにひどい目に会わないとも限りません。そこで、酒フリークな私が、多少そのノウハウを教えられたら嬉しいと思い筆を取りました。

さて、酒にはいろいろ種類がありますね。ビール、ワイン、ウィスキー、梅酒、日本酒、カクテル、... などなど。

初心者の皆さんにお勧めするのは杏酒と白ワインです。「ビールは苦くてちょっと。。」などというひとでも杏酒はジュース感覚でかなりいけると思います。最近は大抵のコンビニにも置いています。ワインは非常に深いのですが、とりあえず初心者の方にはドイツ製の白ワインをお勧めします。甘くて飲みやすいですよ。

ここで注意して欲しいのですが、赤ワインと白ワインは全然違うものです。赤ワインは大体常温で飲みますが、白ワインは冷やして飲みます。また、味わいも赤ワインはこくがあるものが多いですが、白ワインには軽い感じのものが多いです。アルコール度数は赤ワインのほうが若干高めですね。まずは白ワインから入るとよいと思います。

ワインについて一つ忠告すると

安すぎるワインは激まずい

ことです。最低でも 750ml で 700 円くらいするもの以上でないと大抵の場合は残念な思いをすることになるでしょう。逆に、1500 円くらいだせば、相当美味しいワインが楽しめます。

---

[9] 2 面のサーフェスのロックには 100ms 以上かかる。

[1] 素粒子理論やってます。物理を語りたい人や最先端の研究を知りたい人は個人的に連絡をどうぞ。大歓迎ですよ。

「俺は高校時代からビールに親しんできたぜ！」という頼もしい中級者の方には日本酒やウィスキーなど多少度が高いものをお勧めします。さすがに高校時代に、それなりの酒場に入って日本酒を熱燗で頼むことはなかなかできないでしょうから、是非美味しい日本酒を味わって下さい。ウィスキーに関して一つだけ忠告すると

コンビニによくあるサントリーのウィスキーはまずすぎるのでやめとけ

ということです。あれを飲んで「ぐぉー。ウィスキー激まずい。こんなの二度と飲むか」などと思うことのないようにしましょう。比較的安価で美味しいものとしては four roses(バーボンです) などがあります。スコットランドなど本場から輸入したものがお勧めです。また、カティーサークなんかも美味しいです。まずはロックで楽しんで下さい。その後はオレンジジュースで割るとかコーラで割るとかいろいろと自分で工夫しましょう。

上級者の方には私は何も言えません。どうぞ自分の酒世界を切り開いて言って下さい。

次に、ひどく酔っ払ってしまったときの対策についてです。

だれしも一度は悪酔いすることがあるでしょう。酒を飲み始めた頃は自分の限界が分からないのでついつい飲みすぎてしまって吐いてしまう、ということもあるとおもいます。そんなときはどうすればよいのでしょうか？

答は水や烏龍茶などで大量に水分補給することです。アルコールを分解するには水分が必要です。ひどく酔っ払った人をそのまま放置していると、本人では水を取ることすら出来ない場合非常に危険な状態になります。無理をしてでも回りの人が大量に水分を補給させましょう。

そして吐きそうな場合は当然トイレにつれていきましょう。吐いてしまえば大抵の場合かなり楽になれます。(逆に吐けないとつらいです)

あと、自分の限界を超えて飲まされそうになったときは、きっぱりと断りましょう。TSG のコンパでは酒を飲まされるということは絶対にありませんが、他のサークルではあるかもしれません。もちろん自分の許容範囲であるならよいですし、吐くくらいですむならまだ可愛いものです。しかし救急車を呼ぶはめになったり最悪の事態になることも考えられます。きっぱりと「お酒弱いのでもう無理です」と言いましょう。あるいは酔っ払って意識のない振りをしましょう。

以上で私の酒講座は終わりです。皆さんの酒ライフの一助となれば幸いです。よい酒人生を送って下さい。

TSG 用語の基礎知識 2001 年度版

ぶほ編集部

おまけの用語集です。もともとは「そんなに原稿が集まらないだろうから、水増し用の原稿でもつくるか」みたいな軽いノリでつくっていたのですが、蓋を空けてみれば結構な重量級の原稿ばかり。この原稿を掲載しようかどうしようか迷ったのですが、もったいないので載せることにしました。

用語の説明は笑って読み飛ばして頂けると幸いです。では、どうぞ。

あいとーく【italk】　[名サ] ネットワークを利用したチャットシステム。ネットワークに繋がっていれば、リアルタイムに会話（ちゃっと）ができる。同様のものに IRC があるが、思想も実装も全く違う……らしい。サーバソフト・クライアントソフトの殆どは、TSGer が開発・保守している。一応、**裏活動**という位置づけだが、これがメインの活動だと思っている人も少なからずいるようだ。現在 TSGer が入り浸っているサーバーとしては、**main** と **akane** がある。（→ **あかね**、**めいん**）

あかね【akane】　[名] (1) 女子につけられる名前の一。(2) 色の名前。
(3) **italk** サーバーの一。主に 97 年以降に入学した TSGer がここで活動している。管理者は 97 年入学の**ぱら**さん。

あきはばら【秋葉原】　[地名] (1) 千代田区外神田一帯の名称。電気街で世界的に有名。
(2) TSGer、特に本郷近辺にいる人がよく出没する一帯のこと。
(3) [限定的用法] ゲーマーズ。
(4) [限定的用法] CURE MAID CAFE。

あびきょうかん【AVI 叫喚】　[名](1) ハードディスクが AVI ファイルで埋もれること。
(2) ダウンロード速度＞見る速度、のこと。
(3) エンコード速度＜放映量、のこと。

ありあけ【有明】　[地名] (1) 港区台場の近辺の地域。ゆりかもめの終点。
(2) 夏と冬のある一時期に、なぜか大量の人間が集まる地区。おそらくあの近辺のレストランなどはこの現象のためにつぶれずにいるのであろう。

うぃんどうず【Windows】　[商品名](1) OS の一。
(2) Microsoft 社製の再インストール教育ソフト。

	(3) マシンが落ちたときに責任転嫁する対象。「— 2000 は以前のものに比べてだいぶましになっているが、— Me はやはり屑だよね。」
えーでぃーえすえる【ADSL】	[名](1) Asymmetric Data Subscriber Line の略。国の IT 政策により急速に普及が進んでいるが、その恩恵を受けているのは IT 産業ではなく、むしろ**廢人**が殆どである。 (2) 廢を運んでくる魔物。
えむぺぐ【MPEG】	[名] Moving Picture Expert Group の略。動画規格の一。
えんこーど【エンコード, encode】	[名サ] (1) データを一定の規則に基づいて符号化すること。データの圧縮や暗号化などがこれにあたる。MPEG などの様々な規格がある。 (2) TV アニメなどを **MPEG** などの動画ファイルにする廢活動のこと。**白山**近辺で盛んに行われている。(→ **廢**、**廢活動**、**廢データ**)
おーじーえる【OGL】	[名] (1) 株式会社オモイカネにて開発されている Linux ディストリビューション。Omoikane GNU/Linux の略。 (2) ほぼ Zepto 一人の手によって開発されているのは秘密である。 (3) 利用者にナデシコファンが多いという噂であるが、真相は定かではない。
おもいかね【オモイカネ】	[名] (1) TSG の OB が経営する会社。Zepto や菊さんがここでアルバイトをしている。 (2) 社名の由来は推して知るべし。
がっしゅく【合宿】	[名] (1) 数日間、グループで行動を共にし、親睦を深めあう行事。 (2) 観光地で普段通り廢活動をすることを目的としたイベント。もちろん観光もする。かつてはスキー合宿なども行われていたが、最近は夏合宿一本になっている。
きほんてきはいじんけん【基本的廢人権】	[名] 最近制定された基本的権利。具体的には 1.5Mbps 以上の回線を自由に使用できる権利のこと。
くみこむ【KUMIKOM】	[名]20 年ほど前に TSG が開発したコンピューター。これを契機として、TSG はコンピューターサークルとなり、平安京エイリアンという不朽の名作を開発したのである。なお、取扱説明書及びハンドブックが最近発掘された。(→ **平安京エイリアン**)
くみこむさま【KUMIKOM さま】	[名] 305 にあるご神体。神棚の上に安置されている。願掛けをすると開発能力が上がるかもしれない。(→さんまるご)
くみこむしょう【KUMIKOM 賞】	[名] 駒場祭での業績に対して、OB 有志から送られる賞。副賞がユニークである。(→ **駒場祭**)

くらつく【くらつく】 crack(英) と同義。**菊**さん、**HASM** さんの好むこと。

げーまー【gamer】 (adj.) comparative degree of 'game'.
(n.) [*pl*.-s] the person who plays game.

こけし【こけし】 [名] (1) 山形県の伝統工芸品。木製の女性の人形。同様のもの（偽物）にマトリョーシカがある。
(2) 今年度副部長が愛してやまないもの。
(3) 今年度副部長を指す呼称。

こまばさい【駒場祭】 [名] (1) 数日にわたって部員を拘束するイベント。
(2) 自作プログラムやプログラマを展示するイベント。プログラマを展示するのは最終手段である。

さんまるご【305】 [名] TSG の部室のこと。部室が学生会館 305 右奥にあることに由来する。

さる【猿】 [名] (1) 霊長類の動物の一。
(2) 何かをせずにおられない人。中毒。「Italk –」。
(3) italk サーバーに四六時中接続している人、及び italk なしでは生きていけない人のこと。

しす【氏す】 [動サ変] **氏ぬ** と同義。

しぬ【氏ぬ】 [動ナ変] (1) 死ぬ、の **italk** 的表記。
(2) 毎週水曜日深夜 24:45〜25:15 までの間に、テレビ東京をつけること。

じんけん【人権】 [名] (1) 人間の持つ基本的権利。理想論としていえば、「すべての人間の」とすべきであろうが、往々にして侵される運命にあるもの。
(2) **てつ**にはないもの。
(3) **だめ**にもないもの。ただし、だめは**基本的廢人権**を持つ。

せんようせん【専用線】 [名詞] 常時接続されている通信回線のこと。かつては個人で利用できる様なものではなかったが、近年では Flet's ISDN や Flet's ADSL の普及に伴い専用線に近い環境が普及してきた。TSG での専用線（及び疑似専用線）普及率は極めて高い。

だめ【だめ】 [形動ダ] (1) 廢のような。ヲタクのような。
[名] (2) 周囲の人から見て著しく残念に思われる人・ものを指す。

でいむ【dame】 [名] (1) 公爵夫人。(2) **だめ**と同義。

ていれべるくりあ【低レベルクリア】 [名] (1) **やりこみ**の一。ゲームを、なるべくレベルアップせずにクリアすること。

(2) ４学期終了時に、平均点 50.0 + ε 点の極限をめざすこと。TSG の伝統。

てつ【てつ、鉄】 [名] (1) 元素の一。Fe。原子番号 26。
(2) 鉄道について深い知識を有する人のこと。一例として**てつ先輩**・**をやてつ**などがある。

なつがっしゅく【夏合宿】 [名] 夏に行われる合宿の意。(→ 合宿)

はい【廢】 [名] 廃人がやるようなもの、廃人的なこと。廃人そのものを指すこともある。

はいじん【廢人】 [名] (1) 薬物などに耽溺した人のこと。
(2) 「だめ」なものを愛好する人のこと。「オタク」とほぼ同意義。(→ **だめ**)

はいかつどう【廢活動】 [名] 廢人がやる活動のこと。代表例として、**エンコード**・**莫大転送**などがある。

はくー【ハクー】 [名] (1) hack(英) と同意。(2) テキスト RPG の傑作、NetHack のこと。
(3) プロフェッショナルでなければやらないような、ディープな事。
(4) 世の中の新聞には、**くらつく**と同意で使われることが多い。非常に遺憾である。

はこね【箱根】 [地名] (1) 神奈川県箱根町。温泉、駅伝で有名。
(2) 今年の夏合宿における TSG の営巣地。
(3) 将来首都が移転される予定であるかもしれない。地下に空洞があるかもしれない。

ばくだい【莫大】 [形動ダ] (1) 量・数などが著しく多い・大きいこと。
(2) （回線代が）高いこと。
(3) （データの量が）著しく多いこと。

はんかくかな【半角かな】 [名] (1) 半角カナの間違い
(2) すでに過去の遺物。

はんかくかな【半角カナ】 [名] **italk**（特に **あかね**）で最近よく使われる言語の一。「ラメーン」「マターリ」など数々の名詞・形容詞が存在するが、奇妙なことにこの言語には動詞が存在しないようである。

ふく【福】 [名] (1) 好ましい状態のこと。「鬼は外、–は内」

(2) 博多豚骨風のラーメン屋「福のれん」のこと。また、今年度ライブラリアンがよく出現するといわれるラーメン屋。3ヶ月ごとに一週間ほど替え玉（面のお替り）が無料になる期間があり、この期間中はよく infinite 替え玉オフが開かれる。(→ **ラーメン**、**ラメーン**)

ふくる【福-る】　福のれんに行くこと。（→ **福**）

ぶちょ【ぶちょ】　[名]「部長」の略。

へいあんきょうえいりあん【平安京エイリアン】　[商品名] (1) 1979 年発表の名作ゲーム。TSG が企画・制作したというのは事実である。
(2) 305 において最大の床面積を占めているが、最近はもっぱら机として使用されている。

へいしゃ【弊社】　[名] (1) 一人称の所属する会社というほどの意。
(2) 飯田橋駅から歩いて 15 分、合資会社インセプトのこと。

ぼうしゃ【某社】　[名] (1) とある会社というほどの意。
(2) 駒場の裏に建っている「オモイカネ株式会社」のこと。Omoikane GNU/Linux（略称 **OGL**）を開発している。（→ おもいかね）

またーり【マターリ】　[感動詞] まったりとした。

めいん【main】　[名] (1) 主。
(2) `int main(int, char*[]);`(C) あるいは `public static void main(String[]);`(Java) のこと。
(3) **italk** サーバーの一。主に 96 年以前に入学した人がここで活動している。管理人は OB の**あぶら**さん。

めったくり【めったくり】　[企業名] ほぼ**ADSL** 専門のインターネットサービスプロバイダ「東京めたりっく通信」の俗称。

やりこみ【やりこみ】　[名] (1) ゲームなどを、熱意を持ってその隅々まで堪能すること。
(2) 昨年度副部長の得意技。**低レベルクリア**・リアルタイムアタックなど、まさに人外としか思えない所業を行うこと。

ゆーえすびー【USB】　[名] (1) Universal Serial Bus の略。近年はやりの周辺機器とのインターフェース規格。電源をコネクタからも供給できる。
(2) 6 倍コネクタ。＞ Zepto
(3) 305 のマシン名の頭文字を並べるとこうなる。

らーめん【ラーメン】　[名] (1) 中華料理の一。文化の極み。古来より多くの人々を魅了し、「美味しいラーメン屋を見つけることは人生の悦楽」など数多くの名言を遺している。(2) ラメーンと同意。

らめーん【ラメーン】 [名] ラーメンのこと。(→ **半角カナ**)

りぼんのきし【リボンの騎士】 [名] (1) 故・手塚治虫氏の代表作の一。(2) 集英社発行の雑誌「りぼん」を愛好する人のこと。

れい【令、麗…】 [人名] (1) 女子につけられる名前の一。(2) 男子につけられる名前の一。(3) 猫。

編集後記

- ★ ううう、これを書いているのが 5 月 17 日の午前 5 時 3 分。眠みゅい。
- ★ っていうか、Tossy-2 の原稿を直すのに 3 時間ぐらいかかったのは一体どういうことじゃよ。ううう、LaTeX209 形式禁止。っていうか大禁止。
- ★ っていうかみなさん何で英数字を 2 byte 文字で書きますか。2 byte 英数字大禁止。
- ★ もう頭が回らないじゃよー。うー、ばたん。
- ★ と思ったら、部室の dviout が機嫌悪くして印刷させてくれないじゃよ。
- ★ 今日（5 月 18 日）の印刷は諦めるしかないのぉ。
- ★ そして今は 5 月 19 日。明日こそ印刷だ〜。今度は完璧だぜ。PDF ファイルも作って行くしな！………これでプリンタが壊れてたらどうしよう（汗

理論科学グループ　部報 第 235 号

2001 年 5 月 20 日　発行

発行者　西田健志

編集者　會田雄亮

発行所　理論科学グループ

〒 153–0041　東京都目黒区駒場 3–8–1

東京大学教養学部内学生会館 305

Telephone: 03–5454–4343

理論科学グループ部報 第 235 号

— 新歓コンパ号 —

2001 年 5 月 20 日

THEORETICAL SCIENCE GROUP